Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.12, 2012

Quality Flaws: Issues and Challenges in Software Development

P Mohan[1*] A Udaya Shankar[1] K JayaSriDevi[2]

1. Hyderabad Business School ,GITAM University, Rudraram, Patancheru Mandal, Hyderabad, AP 502329, India
2. Jawahar Navodaya Vidyalaya,West Godavari,AP 534001,India

* E-mail of the corresponding author: pmohan901@gmail.com

Abstract

A statement "Prevention is better than cure" for illnesses in medical sciences also applies to the software development life cycle in terms of software defects. A defect is a deviation from actual functionality of the application in terms of the correctness and completeness of the specification of the customer requirements. Defective software fails to meet its customer requirements leading to the development of applications with poor quality. Quality is a top priority in every enterprise these days. Organizations struggle in a treadmill race to deliver quality software to stay ahead with new technology, deal with accumulated development backlogs, handle customer issues as software teams work as hard as they can make their organizations stay alive and competitive in the market place. Software companies face an immense pressure to virtually release a bug-free product or a software package. The culture of an organization is a critical success factor in the efforts of process improvement. The paper aims at assessing quality as a function for monitoring and measuring the strength of development processes and any successful application development enterprise requires an unambiguous understanding of customer expectation and maximizing participation of customers in the development activities thereby ensuring that people involved in development activities do the right thing and do the thing right for delivering high quality software.

Keywords: Software development, process improvement, software defect, bug-free product, software package

1. Introduction

The process of developing new software is a time taking effort. Software development firms are adopting new methodologies and refining their techniques to mediate risks of developing software products that meet their client requirements. However development process in most of times proceeds with a general risk such as failure to meet the system's specified requirements and sometimes not meeting the requirements at all leading to "software crisis". (Edsger Dijkstra,The Humble Programmer (EWD340), Communications of the ACM, Michael Jackson, "Engineering and Software Engineering" in S Nanz ed, The Future of Software Engineering, Springer Verlag 2010; Michael Jackson, Problem Frames: Analyzing and Structuring Software Development Problems; Addison-Wesley, 2001) The primary goal of software developers is to produce quality systems and products that meet needs of the end user. Thus "Quality" is defined in terms of customer as meeting expectations and standards of the customer. (Kitchenham and pfleeger[6])

2. Statement of the Problem

"Software development" process involves information translation, usually from simple end user requirements to a complex application. As the translation process involves human-based activities, mistakes

are likely to occur in more number. To develop bug-free software, several mechanisms are needed to identify and prevent the errors. Software firms must focus more attention on quality for their everlasting success. This is specifically more important at the early stages of development. This reduces the chance of propagating the faults to subsequent stages in development.

3. Research Objective

The key objective of this study is to explain the importance of quality in software development process resulting in enterprise success. The study covers aspects related to factors that limit the scope of development leading to developing systems that have surprising negative affects where the root cause being the relevant requirements of the stake holders were not being considered and also addresses the issues with some applicable suggestions.

4. Methodology

Software development organizations follow a systematic and well disciplined manner of developing various platform based applications called '*Software Engineering*'. Software Engineering is a layered technology. The timely development of computer software must establish a process framework for effective delivery of software technology. This software process must then form the basis for control of software project management, define the way technical methodologies are applied and finally specify how the work products are produced

4.1 The Process Framework

Application development in any IT industry relies on a process framework that applies to all software projects. Each framework activity is populated by a set of software engineering actions resulting in production of software engineering work product. A generic process framework for a software project incorporates the following tasks (Software Engineering: A Practitioner's Approach-Roger S Pressman):

Communication(Requirements Gathering): It involves communication and collaboration with the customer and focuses on gathering end user requirements and corresponding review activities for their correctness and completeness.

Planning (Analysis and Design): It establishes a plan for software development work, describing the technical tasks to be conducted focusing mostly on the work schedule.

Construction (Implementation): It involves the activities of coding (program generation) and testing of generated code to uncover errors.

Deployment (Maintenance): An activity where the developed work product (usually an application build) is delivered to the customer as a complete entity or partially completed increment. The application in its due course of development passes through all these stages before finally being handed over to the end user. However project requirements change continually but the impact of change varies with the time at which it is introduced. When requirement changes are requested early, but good application architecture must easily accommodate changes with the underlying design being flexible.

4.2 Common Problems in Software Development Process

With the increasingly becoming complex needs of the end user in business environment, IT firms commonly face the following five problems during application development (riceconsulting.com):

Miscommunication: Customers of the work product tend to assume and misinterpret a lot of things when communicating business requirements, causing a widespread miscommunication of information during all the phases of software development.

Complexity of Software: Rapidly changing business needs of the customer results in development of enormously complex software that cannot be fully understood.

Coding Errors: Application developers are prone to making errors resulting in the generation of software bugs.

Dynamic Requirements: Change in customer requirements during development process certainly impacts software functionality. A system with changing requirements demand additional functionality in the existing system which may affect existing modules in unforeseen ways as interdependencies between modules can make the system error prone.

Unrealistic Schedule and Inadequate Testing: Cramming too much of work in too little time makes problems inevitable and also no one knows whether or not the software is good until customers complain.

5. Analysis

Errors in software requirements and design documents are more frequent than the coding errors. Application development accomplished by a modeling activity in the process framework comprises of two basic software engineering actions such as '*analysis'* and '*design'*. *Analysis* includes a set of tasks such as requirements gathering, negotiation, specification and validation. *Design* encompasses work tasks that include low-level design, high level design, architectural and interface design.

5.1 Origin of Software Defects

A number of studies made by software communities reveal that most failures in software products are due to errors in requirements and design phases. Common reasons for introduction of defects into software include the following:

Incomplete or Erroneous Specification (IES): Most often before the development process begins, customer requirements may be ambiguous because of the inaccurate specification of what actually is needed. Recommendations to resolve such inconsistent requirements specification requires a formal review.

Questionnaire for requirements review include:

1. Are the requirements correct?
2. Are requirements complete?
3. Are they time bound to enable successful project completion to meet timelines?
4. Are they randomly changing (agile in nature)?

Misinterpretation of Customer Communication (MCC): The cost of misunderstanding seems too big to

ignore as often the result of a misinterpretation doesn't come to the forefront until some responsive action such as matching to the listener's style and adjusting to it accordingly takes place.

Intentional Deviation from Specifications (IDS): User interface designers strive to develop effective and ultimate interface. But with a commercial development environment in place, resource constraints affect the final product design. Thus application designers need to revise their perception of the end user specification to provide the best implementable user interface.

Error in Data Representation (EDR): An application developed for business processing tasks interacts with varied kind of input data. Improper or ambiguous data requirements may lead to development of application controls that accept invalid input both related to the type and size of the data.

Error in Design Logic (EDL): Design errors may lead to development of a product that goes against the requirements specification. Logic verification tools need to be often used during the development process, to verify a system in terms of its functional specification.

Incomplete or Erroneous Testing (IET): If the test processes are defective, testers will create as they conduct software design with incomplete or erroneous decision-making criteria resulting in added bugs.

5.2 Software Requirements Analysis

It is important to analyze requirements in place to ensure customer needs are correctly translated into product specifications. An iteration of interactive sessions with the customer can help development people in understanding of the actual requirements. The figure below depicts that errors in the requirements and design phase contribute to 64 percent of total defect costs.

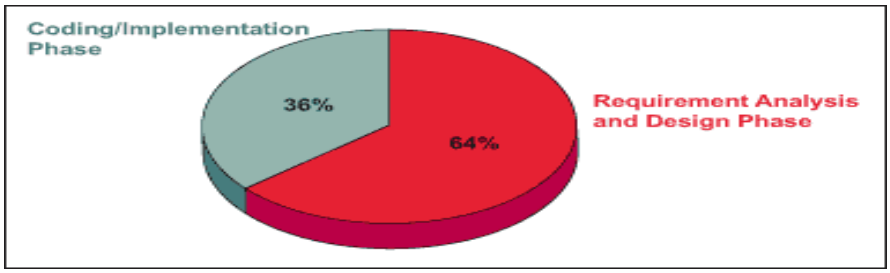

Figure 1. Origin of Defects

Source: Crosstalk, Journal of Defense Software Engineering

5.3 Root Cause Analysis

The root cause analysis of a defect majorly focuses on the following three key principles:

Reduce defects to improve quality: Analysis process should lead to changes that help early detection of defects and their prevention.

Apply Logical Expertise: Members of software engineering members need to provide suggestions for how to avoid defects in the future.

Identify Systematic errors: Mistakes tend to be repeated contributing in a large portion in typical software project. Identifying such defects can have a big impact on quality.

5.4 Cost Impact of Software Defects

A primary objective of the software development organizations is not only to develop work products in line to the requirements specification, but also to focus on finding and reviewing errors during the development process so that they do not become defects after release of the software. This makes possible, the early discovery of errors so that they do not propagate to the next step in the software process. Design activities introduce 50 to 65 percent of defects during the software process. Review processes need to be established during development process activities to substantially reduce large percentage of errors there by reducing the cost of subsequent activities in the software process. Defects introduced during the requirements and design phase are more severe and difficult to remove via testing. They need pre-test reviews. The table below shows different phases of the software development life cycle and percentage of defects introduced in each phase.

Table 1. Defects Introduced into Different Phases of Software

Software Development Phases	Percent of Defects Introduced
Requirements	20 Percent
Design	25 Percent
Coding	35 Percent
User Manuals	12 Percent
Bad Fixes	8 Percent

Source: Secondary data available at Computer Finance Magazine

5.5 Early Defect Detections

To develop good quality software, it is necessary for testing teams to have an early defect detection method. This also supports for defect prevention by involving a structured-problem solving strategy in identifying, analyzing and thus preventing the occurrences of defects. The figure below shows the cost to fix defects during various phases of the software development cycle.

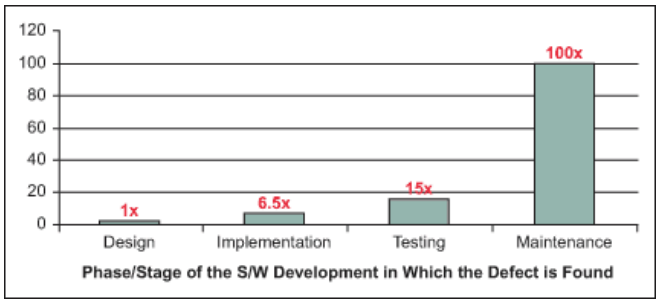

Figure 1. Costs to Fix Software Defects

Source: Available at IBM Systems Science Institute

The Table below illustrates cost impact of early error detection.

Table 2. Cost to Correct Errors During Various Phases of Development

Type of Error	No. of Monetary Units to Correct
During Design	1.0
Before Testing	6.5
During Testing	15
After Release	60-100

Source: Secondary data taken from Software Engineering: A Practitioner's Approach Sixth Edition.

6. Significance of Software Quality Assurance (SQA) in Software Development

Organizations incur a lot of money requesting IT firms to develop quality software for handling a specific business need. Because of this valuable nature of the software, development processes need to maintain quality of work in various phases to ensure quality in the final product. This concern led to the introduction of SQA activities that are aimed at developing a sound software development methodology to produce quality software. Software quality assurance focuses on two different constituencies:

1. Application developers who do the technical work
2. SQA groups, responsible for quality assurance planning, oversight and record keeping.

6.1 SQA Activities

SQA groups assist software teams in achieving a high quality end product. Development process in the organizations must encompass a set of SQA activities to address quality assurance planning, analysis and reporting. Activities to be performed by independent SQA groups include:

Preparing an SQA project plan: The SQA plan must identify the evaluations to be performed, standards applicable to the project, procedures for error reporting and tracking, and amount of feedback provided to the project team.

Participation in software process description: The process framework selected by the project teams must be reviewed by the SQA groups for compliance with organizational policy.

Ensure work deviations in process and product are documented: The SQA group must review work products, track deviations, verify that corrections have been made and periodically report results of the work.

Record Noncompliance: Non compliance items must be tracked until they are resolved.

7. Top Recommendations for Addressing Quality Issues

At the start of any new project, there exists a great deal of uncertainty which forms the basis for the development of defective software. Attempting to remove such uncertainty at the beginning of a project development work sometimes results in a huge blunder in a traditional process framework for application development. An approach to eliminate uncertainty is to focus initially on removing the chaos on what will

be developed by making the development teams do a lot of upfront thinking about what the users actually expect.

7.1 Embedding Procedures into Software Development Process

With the consent of the management and commitment from the development team, a plan of action can be developed for propagating the modification of the existing processes or the introduction of new ones. Some of the activities in this phase may include the following:

1. Severe defects and their analyses should be mentioned on monthly status.
2. Meetings should take place to make the team aware of the symptoms and solutions of the systematic errors.
3. Learn from the root cause analysis of previous projects and use it as a base line for future projects.
4. Monitor the defect prevention progress.

Apart from the above procedures, common solutions to eliminate faults during application development include:

Iterative Development: Choose teams that work iteratively and attempt to reduce uncertainty by making them understand that it is impossible eliminate all uncertainty about what a product is to be at the start of the project.

Prototyping: Active involvement of the customer during development process plays a vital role for ensuring quality in the product. This can be achieved by showing the parts of product under development to the customer.

Pre-user acceptance: Before actually deploying the work product, feedback needs to be collected, project plans must be adjusted and opinions need to be refined.

7.2 Defect Prevention Principle

To address the defects raised during development process and to prevent them for further cycles, test engineers follow a defect prevention cycle as illustrated in the figure below:

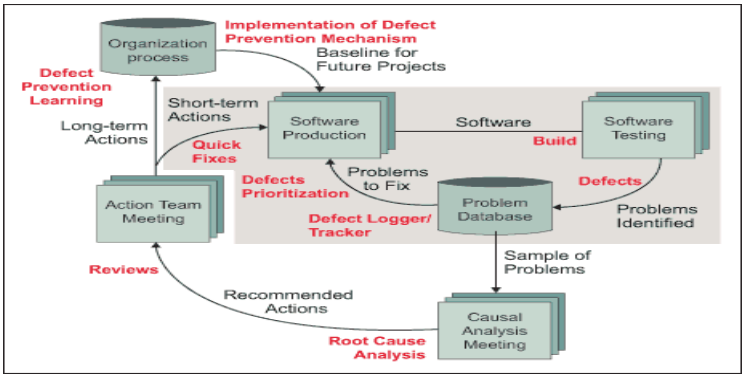

Figure 2. Defect Prevention Cycle

Source: Secondary data available at IEEE Software Productivity Consortium

The gray colored boxes in the above figure represent handling of defects which include activities like-

defect detection, documenting and analysis of defects, arriving at short-term solutions. Defect prevention practices enhance the software developer's ability learn from the errors. Common benefits of defect prevention methodology include:

1. Significant Reduction in number of defects and their severity.
2. Improved product quality
3. Reduced rework effort
4. Better communication among the project team

8. Conclusion

The success of an engineering approach relies on organizational commitment to quality Although software development is complex, and is error prone, many problems that are faced during the development of a work product can be tackled, by adopting a good software development process such as establishing auditing and reporting function of management to compare the work products with specified and measurable standards enabling project management to gain the insight and confidence of product quality. From our discussion, it's apparent that good process frameworks are essential. The software industry is still learning, about good processes for software development. Adoption of quality philosophies like TQM, Six Sigma enable organizations to focus on a continuous process improvement leading to the development of more effective software applications.

9. References

http://www.sdresearch.org/Software%20Tools/Reducing_software_failures.pdf

http://www.mountaingoatsoftware.com/articles/23-the-certainty-of-uncertainty

http://www.riceconsulting.com

Rick Hower's "Software QA and Testing Resource Center" (Source: www.software qatest.com)

Software Testing and Quality Assurance (Source: http://www.software-quality-assurance.info/)

Any software code will have bugs- Microsoft (Source: http://www.ciol.com/content/ news/repts/102100302.asp)

The Software Crisis (Source: http://www.unt.edu/benchmarks/archives/1999/july99 /crisis.htm)

Roger S Pressman, Software Engineering:A Practitioner's Approach.

Carlo Ghezzi,Mehdi Jazayaeri,Dino Mandrioli-Fundamentals of Software Engineering.

Robert L.Glass, Facts and Fallacies of Software Engineering.

Yingxu Wang,Software Engineering Foundations: A Software Science Perspective.

First Author: P.Mohan is living in Hyderabad. He was born on 10 August, 1981. He obtained his M.Sc in Computer Science in the year 2003 from Osmania University, Andhra Pradesh, India. He had also published various research papers in National and International Journals. He has 10 years of academic teaching experience. At present he is working as Programmer at Hyderabad Business School which is a constituent of GITAM University (www.gitam.edu) Visakhapatnam, Andhra Pradesh, India.

Second Author: A.Udaya Shankar is living in Hyderabad. He was born on 15 June 1975. He obtained his MBA in the year 1998 from Sri Venkateswara University, Andhra Pradesh, India. He obtained his doctorate from Andhra University. His research thesis is on "Interface of CRM, Brand Equity and Shoppers'. Behaviour in Organized Retailing-A Comparative Study of select Retail Marts in Greater Hyderabad". He had also published a research paper in an International Journal and authored a book titled "Entrepreneurial Management". He has 14 years of academic teaching experience. At present he is working as Associate Professor in Marketing at Hyderabad Business School which is a constituent of GITAM University (www.gitam.edu) Visakhapatnam, Andhra Pradesh, India.

Third Author: K.JayaSriDevi is living in Eluru. She was born on 03 June 1980. She obtained her M.Sc(CS) in the year 2004 from Achary Nargarjuna University, Andhra Pradesh,India. She obtained her M.Tech(CS) in the year 2010 from Achary Nargarjuna University, Andhra Pradesh,India. Pursuing **Ph.D** in the area of Image Processing from **Acharya Nagarjuna University** & completed **Pre-Ph.D**. She has 8 years of academic teaching experience. At present she is working as FCSA in Jawahar Navodaya Vidayalaya, WestGodavari, Andhra Pradesh, India